UNIVERSITY OF MICHIGAN ☙ SCHOOL OF INFORMATION
SI 539: Design of Complex Web Sites

Assignment 9 – Ajax and Functional Tests

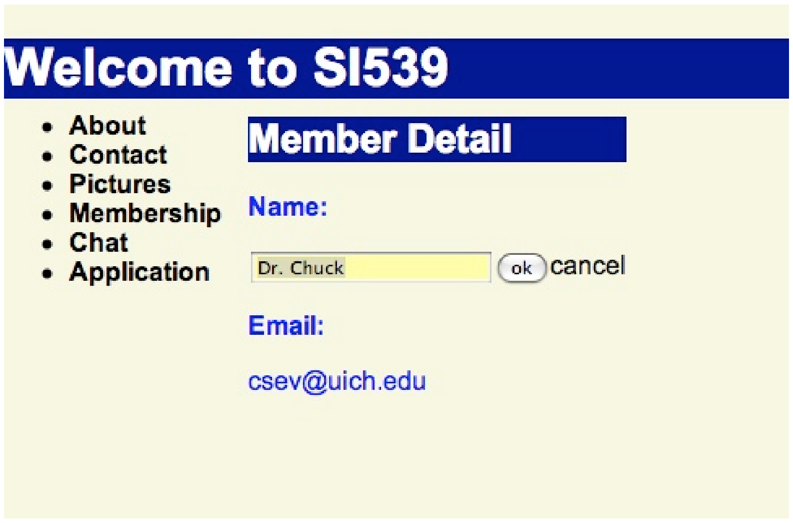
Due Date: Tuesday April 1, 2008 at 11:55PM

In this assignment you will add two Ajax-based features to your application as well as add a series of unit tests to your application.

First make a copy of your previous assignment to begin work,

Add Ajax In-Place Editing to your View

In this section you will add code to your application so that the view allows editing and saving of the member information using Ajax:



You navigate to this screen from the Membership screen by pressing "View". There is no new view - just new functionality in an existing view.

Edit your application template (the file that wraps all of your other rhtml files) and add a tag to include the necessary Javascript Ajax libraries:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <%= stylesheet_link_tag "style.css" %>
 <%= javascript_include_tag :defaults %>
</head>
```

Edit your controller and add the following methods:

```
in_place_edit_for :member, :name

def set_member_email
  logger.info "Doing set_member_email manually"
  item = Member.find(params[:id])
```

```
    item.update_attribute(:email, params[:value])
    render :text => item.email.to_s
  end

  # Make the getters
  def get_member_name
    item = Member.find(params[:id])
    render :text => item.name.to_s
  end

  def get_member_email
    item = Member.find(params[:id])
    render :text => item.email.to_s
  end
```

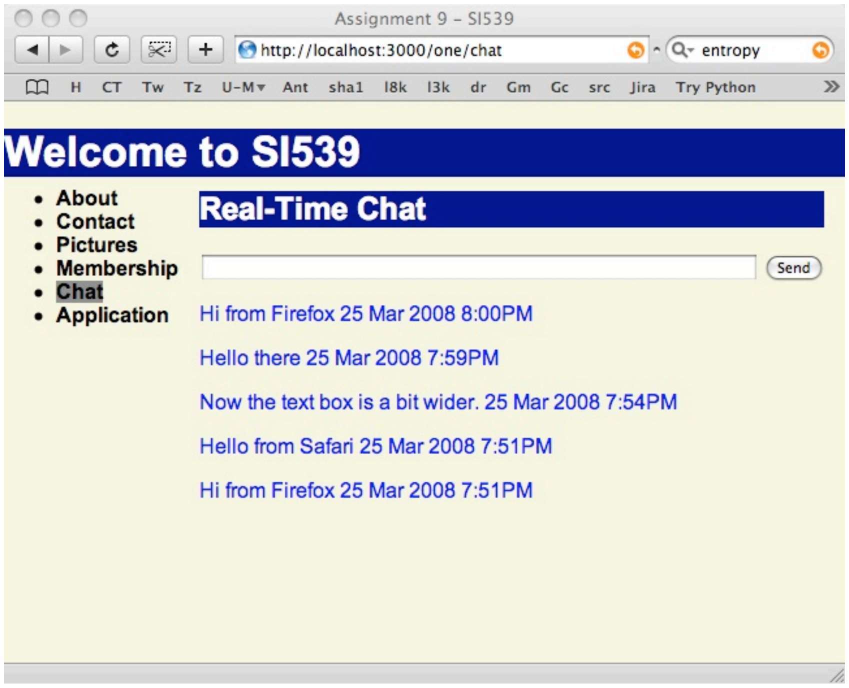
Replace your **view.rhtml** file as follows:

```
<h2>Member Detail</h2>
<p>
<strong>Name:</strong>
</p>
<div  class="inPlaceEditorWidget">
<span class="inPlaceEditor">
    <%= in_place_editor_field "member", "name", {}, {
          :load_text_url => url_for(:action => "get_member_name", :id =>@member)
         } %>
</span>
</div>
</p>
<strong>Email:</strong>
</p>
<div  class="inPlaceEditorWidget">
<span class="inPlaceEditor">
    <%= in_place_editor_field "member", "email", {}, {
          :load_text_url => url_for(:action => "get_member_email", :id =>@member)
         } %>
</span>
</div>
```

At this point you should be able to use your view file and edit either field in place.

Adding Ajax-Based Real-Time Chat to Your Application

Next we will add real-time chant to your application using Ajax. This is done by periodically updating a div as well as simple form to post new chat messages and update the div.



Hi from Firefox 25 Mar 2008 8:00PM

Hello there 25 Mar 2008 7:59PM

Now the text box is a bit wider. 25 Mar 2008 7:54PM

Hello from Safari 25 Mar 2008 7:51PM

Hi from Firefox 25 Mar 2008 7:51PM

First make a new model for your application.

```
ruby script/generate Model chat
      exists  app/models/
      exists  test/unit/
      exists  test/fixtures/
      create  app/models/chat.rb
      create  test/unit/chat_test.rb
      create  test/fixtures/chats.yml
      exists  db/migrate
      create db/migrate/002_create_chats.rb
```

Edit the db/migrate/002_create_chats.rb file and add the following columns to the model:

```
def self.up
   create_table :chats do |t|
     t.column :chatmsg, :string
     t.column :member_id, :integer
     t.column :created_at, :datetime
   end
 end
```

We will not use the member_id in this assignment - but putting it in now prepares you for the next assignment.

Then run the rake command to run your second migration:

```
rake db:migrate
(in /Users/csev/Desktop/teach/a539/w08/rails_apps/assn9)
== CreateChats: migrating ========================================
-- create_table(:chats)
   -> 0.0044s
== CreateChats: migrated (0.0046s) ===========================
```

Remember if you have a problem you can un-migrate with

```
rake db:migrate VERSION=0
```

and then

```
rake db:migrate
```

to re-run the migrations. You should verify that your table was created correctly using the SQLite Browser before you continue.

In your navigation file view file, add an entry for Chat:

```
<%= do_nav_entry("Membership","members") %>
<%= do_nav_entry("Chat","chat") %>
<%= do_nav_entry("Application","join") %>
```

Create a new view file for the chat action as follows:

```
<h2>Real-Time Chat</h2>
<p>
 <% form_remote_tag :url => 'chatcontent', :update => 'chatdiv' do -%>
  <input type="text" size="60" name="chatmsg"/>
  <%= submit_tag 'Send' %>
 <% end -%>
</p>
<%=  periodically_call_remote(:url => 'chatcontent',
     :frequency => '3', :update => 'chatdiv') %>
<div id="chatdiv">
<%= image_tag "ajax-loading.gif" %>
Loading ...
</div>
```

Find and download an animated gif form the Internet. Hint: Search Google Images for "Ajax Loading" and save the file into your public/images and place its name in the **image_tag** above.

Create a new view file called **chatcontent.rhtml** as follows:

```
<% for chat in @chats %>
<p class="chattext"><%= chat.chatmsg %>
<span class="chatdate">
```

```
<%= chat.created_at.strftime("%e %b %Y %l:%M%p") %>
</span>
</p>
<% end %>
```

Edit your controller and add the following method to the controller:

```
def chatcontent
  if request.post? and params[:chatmsg] != nil
    logger.info "Chat"
      ch = Chat.new
      ch.chatmsg = params[:chatmsg]
      ch.save
  end
  @chats = Chat.find(:all, :order => "chats.created_at DESC",
     :limit => 5)
  logger.info "We found #{@chats.size} chats"
  render :action => 'chatcontent', :layout => false
end
```

At this point your chat should start working. To test the chat use two browsers open at the same time - you should test the asynchronous updating - send a message in one browser and wait three seconds and it should appear in the second browser.

Functional Unit Test

You need to add a functional unit test to your application as well. In the file,

test\functional\one_controller_test.rb

Add at least five unit tests as follows:

A test to retrieve the main page and assert success (test_one below)
A test to make sure that navigation highlighting is working between pages (test_index and test_join below)
A test that makes a POST and verifies success (test_join_success below)
A test that makes a failed POST verifies failure (test_join_error_01 below)

If you have not been following my examples (i.e. you have your own site) you will need to adapt the examples below - but the changes should be relatively straightforward. Consult this week's lecture for additional explanation of this information. The material in bold is the additional material - you should not have to make any changes to the code not in bold that is provided by ruby script/generate for your controller.

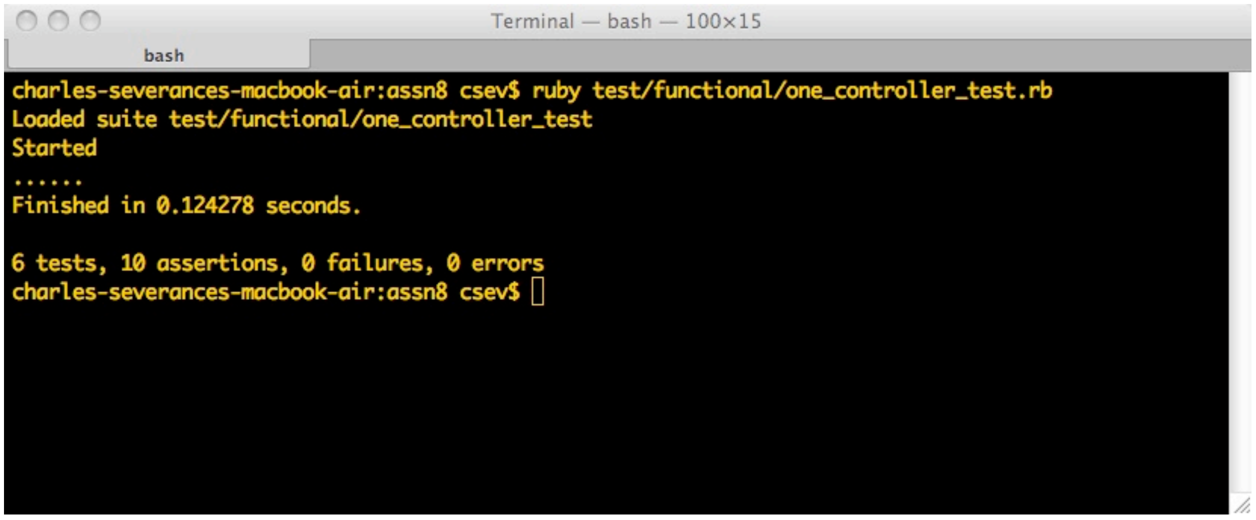
```
require File.dirname(__FILE__) + '/../test_helper'
require 'one_controller'

# Re-raise errors caught by the controller.
class OneController; def rescue_action(e) raise e end; end

class OneControllerTest < Test::Unit::TestCase
```

```
  def setup
    @controller = OneController.new
    @request    = ActionController::TestRequest.new
    @response   = ActionController::TestResponse.new
  end

  def test_one
     get :index
     assert_response :success
  end

  def test_index
    get :index
    assert_select  'a.selected', 'About'
  end

  def test_join
    get :join
    assert_select  'a.selected', 'Application'
  end

  def test_join_error_01
    post :thanks
    assert_not_nil flash[:notice]
    assert_redirected_to :action => 'join'
  end

  def test_join_error_02
    post :thanks, 'yourname' => 'Chuck'
    assert_not_nil flash[:notice]
    assert_redirected_to :action => 'join'
  end

  def test_join_success
    post :thanks, 'yourname' => 'Chuck',
        'yourmail' => 'csev@umich.edu'
    assert_nil flash[:notice]
    assert_response :success
    assert_template 'thanks'
  end

end
```

To test your controller functional test, run this command in your application directory:

```
  ruby test/functional/one_controller_test.rb
```

And verify that your tests sun successfully as shown below:



```
charles-severances-macbook-air:assn8 csev$ ruby test/functional/one_controller_test.rb
Loaded suite test/functional/one_controller_test
Started
......
Finished in 0.124278 seconds.

6 tests, 10 assertions, 0 failures, 0 errors
charles-severances-macbook-air:assn8 csev$ 
```

Hand In

Hand in equivalent screen shots for you application for each of the screen shots shown above in this hand out.

In addition, hand in the following files: **one_controller_test.rb, view.rhtml, chat.rhtml, chartcontent.rhtml**

If your program has a controller with different names for files - just hand in the equivalent files for your application.